BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

measures such as delay time, delay variability, and reliability. These requirements are not presently specified in a flexible manner, though they may vary for different parts of a complex telecommunications application. For example, if a voice-mail system is used to record a voice call between two parties, low delay is important between the human parties but not in the path to the voice mail's storage location.

In addition to specifying the behaviour and quality of service that the application desires from the telecommunications system, optimal use of the telephone system's resources requires it to describe the loads that it will place on that system, for example in terms of bandwidth requirements on communications links and in terms of processing power required in computation nodes. Current systems do not have this capacity.

The complexity of present telecommunications systems software, and the extensive interactions between its software components, makes the development of new features very difficult. As well, telecommunications services have traditionally been provided by large monopolies who employed proprietary equipment that only they had access to. Another complexity is that new services had to be backward compatible to handle their existing clientel.

Software development is therefore limited to a "closed" group of trusted developers, which reduces the talent pool available and shuts out developers with new ideas for niche markets.

Traditional telecommunications does not consider differentiation, but single service. Therefore, telecommunications providers would not be encouraged to offer varied services at a cost reduction to users, for example, reduced quality of voice telephony on Christmas Day, simply to provide additional connections or reduced cost.

As well, small niche markets have gone unserved completely as the cost of developing and implementing the additional products does not net sufficient profits.

Telephony systems as currently implemented comprise "switches" controlled by large computer programs and interconnected by a variety of means, such as optical fibre and coaxial cables. These systems also include computing means to implement such features as conference calling and voice mail. Telephony features, such as voice mail and call forwarding, are implemented by adding code to the programs running the switches and by adding specialized hardware to the telephony network. The features available to particular users are defined in databases

accessed by the switch software, and adding a new type of feature may involve changing these databases together with all of the switch software that uses them, and may also involve purchasing and installing new types of hardware in the network. Specialized software is also used to check the consistency of the features assigned to a particular user, for example, call-waiting and call-forward-on-busy features define different behaviours for the same event; a busy receiver.

Changes to the existing telecommunication networks are therefore very complicated to make. There is a rigid model and hardware structure is difficult to extend. Therefore, existing telephone companies can not offer new features such as high quality voice. As well, existing telephone companies take a long time to bring such features to market.

Users can exercise a small degree of control over their telecommunications by use of software running on their personal computers (PCs). For example, there is currently a Telephony applications programming interface (TAPI) that allows software running on a general-purpose computer to control the switching decisions of a type of switch known as a private branch exchange (PBX). An application programming interface (API) converts a series of comparatively simple and high level functions into the lower level instructions necessary to execute those functions, simplifying control of an operating system. Using Windows APIs, for example, a program can open windows, files, and message boxes, as well as perform more complicated tasks, by executing a single instruction. Windows has several classes of APIs that deal with telephony, messaging, and other issues.

The TAPI consists of a large collection of specialized subroutine calls that allow a user to set up and tear down circuits connecting particular physical devices, including telephone sets and servers for functions such as voice-mail. It also allows the user to define how the system should respond to events such as hangups.

A system known as Parlay and developed by a consortium of companies implements a telephony API that can be used to control the central office telephone switches owned by large telephone companies. This is similar in concept to the use of a telephony API to control a PBX, but security concerns are of prime concern because of the number of telephone users who would be inconvenienced by a failure.

Parlay, TAPI, J-TAPI and similar systems permit third parties a degr e of control over how telephone switches interconnect end users and specialized equipment such as voice-conferencing servers, but do not allow third parties to add

new features such as encryption or voice coding. They are also unable to describe the handling of Internet traffic, and so it is necessary for a distinct system to be used to handle such functions as routing Internet browsing data through computers acting as security firewalls.

Socket mechanisms are widely used to describe connections between applications programs running on operating systems such as UNIX and Windows. It can be used to set up connections between applications programs running on different computers, such that packets of data are passed between them across such networks as an Ethernet or the Internet.

When using a socket to communicate with a process on another computer, the programmer defines one side of a communication but must rely on the administrators of the other computer to have set up the other side. The port number is used by convention to describe the functionality of the program expected.

There is therefore a need for a method and system of providing telecommunication services that are flexible and efficient, and improve upon the problems described above. This design must be provided with consideration for ease of implementation and recognize the pervasiveness of existing infrastructure.

Summary of the Invention

It is therefore an object of the invention to provide a method and system of providing telecommunication services that is flexible and open to modification and improve upon the problems described above.

One aspect of the invention is broadly defined as a method of implementing a communication over a telecommunications network comprising the steps of:

composing the communication in terms of a graph of software building blocks; and

dynamically instantiating the graph of software building blocks at run time.

Another aspect of the invention is defined as a method of implementing an application programming interface (API) for graph-based implementation of telecommunications comprising the steps of: receiving input instructions; and

responding to the input instructions by generating a graph describing the desired functionallity of the communication.

Another aspect of the invention is defined as a method of implementing a communication over a telecommunications network, the communication being

The invention which improves upon the problems described above is a method of implementing a communication over a telecommunications network by composing the communication in terms of a graph of software building blocks, and then dynamically instantiating the graph of software building blocks at run time. This software structure allows the flexible and efficient processing of signals or data streams in communications systems, computer systems and computer networks.

The schematic diagram of **Figure 1** presents a simple example of an implementation of the invention. In this example, the user agent **10** which desires the communication, generates a data structure **12** which identifies the software building blocks and if necessary, configuration data that it requires to perform the communication. This data structure **12** is transmitted to the network **14**, which uses this data structure **12** to assemble the software building blocks in the necessary order and to interconnect them as required. If the software building blocks have been given specification configuration details, then the network **14** assigns those configuration details prior to execution of the software building blocks.

This system is very bandwidth efficient, in that large blocks of software code to perform desired functionality are not transmitted around the network **14**, but comparatively small graph data structures **12** that identify the software building blocks to be executed. The software building blocks may be stored in any accessible location, such as locally, at a local cache or server cache, or at a third party location. Third party locations, may, for example, be identified using an Internet universal resource locator (URL) address. This allows third parties to generate new software building blocks and make them available.

The concept of a software graph refers to functional routines, or filter nodes, being the software building blocks and the graph structure **12** itself describing how to link those building blocks together. Therefore, a call is defined in terms of function flow rather than data flow, and the graph data structure **12** may be as simple as a table of pointers, as long as all of the participants know what the contents mean. The graph data structure **12** could be handled by the network **14** as one or more packets.

From the User's perspective, the invention is embodied as an application programming interface (API) which allows the user to identify desired communication features and parameters, and to generate corresponding graphs. As noted above, an API allows the user to select functions at a high level while the API generates the corresponding low-level software code.

The network **14** has complementary operability which allows it to receive a graph of software building blocks and to dynamically instantiates the graph of software building blocks at run time. As will be described with respect to the preferred embodiment, the network **14** is also able to identify and correct certain inconsistencies in the graph it receives.

In the past, only the telecommunications providers were able to provide new functionality to the system, but the open and flexible model of the invention allows any third party to add new filters to the system. All that is required is knowledge of the standard and freely available specification for input and output ports.

As well, the invention provides for a standard API that provides intercompatibility between users, service providers and third parties designing new applications. Third parties may make these new application freely available, or may obtain financial compensation for their use via known electronic commerce techniques.

The invention accommodates new technology and changing market demands and allows for the continuous addition of new services, simply by addition of new filter nodes. Because filter nodes are defined in terms of their properties, coordination and intercompatibilty of filter nodes may be easily administered. In the preferred embodiment, it is a standard part of the graph design philosophy that filters be universal and can be arranged in any order. This is due to the standard input and output rules, strong typing which avoids misconnection of filters, and the graph itself, to call the filters.

Telecommunications systems in the past only provided a very small number of specific services, but today each successive call may require different functionality. Common variations include requirements for security, quality such as delay, bandwidth and reliability, services such as call forward, call waiting and conference call, and varying requirements for hardware, geography and administration. Implementing all of these features on the existing telecommunications model would require immense quantities of complex code.

While existing telecommunications software must be very complex to allow for such variety in operating modes and user preferences, the invention may, if necessary, handle all of these modes with separate filters. If the variations to a filter are straightforward, they may be included in a single filter node which is configured at

time of execution. Alternatively, if the variations are unmanageable, separate filter nodes may be defined. This allows new features to be added quickly and easily.

The configurability of the invention by addition of new filter nodes also addresses problems in the art such as backward compatibility. In the past, older telecommunications switches had to be re-programmed to add new features. To implement the invention with such new switches, one merely designs a new filter node as an interface to the old switch.

Similarly, the openness and flexibility of the invention also overcomes the limitations of existing socket and API design, provides flexibility to change from one transport medium to another, and allows routers and switches to perform load management that was not available in the past.

Most importantly, the accessibility of this system to all interested parties allows the quick progress of new features, exploitation of niche markets that would not have been addressed by telecommunications companies in the past, and does so in a reliable and efficient manner. The expectation is that the invention will commoditize telephony software in a manner similar to how the personal computer commoditized software that had previously been dominated by mainframe computer providers. Rather than requiring man years for telephone companies to develop proprietary software, the invention allows a single user to spend a few hours creating a new filter to the benefit of thousands of users world wide. The invention also provides such freelance software developers to obtain financial compensation for their efforts using existing electronic commerce and milli-commerce techniques.

Detailed Description of Preferred Embodiments of the Invention

As described above, the invention includes a first step of creating a graph composing a communication in terms of a set of software building blocks or filter nodes, and a second step of instantiating that graph at run time.

At a system level, the invention will be applied to a generalized telecommunications network **14** as presented in **Figure 2**. In this figure, the telecommunications network **14** is presented as including both a public switched telephone network (PSTN) **16** and Internet **18**. Part of the Internet network **18** is shown to comprise an asynchronous transfer mode (ATM) network **20**, but other telecommunication networks are also compatible with the invention including synchronous transfer, integrated services digital network (ISDN), asynchronous

digital subscriber line (ADSL), local area networks (LANs), and wide area networks (WANs). Users may connect to this network **14** by use of hard wired telephones **22** or wireless telephones **24** which connect to the network **14** through base stations **26** of service provider BS-A. Note that these base stations **26** are interconnected via a backbone which may comprise hard wired interconnections, IP or ATM routers as shown, or other similar means.

Similarly, computers **28** could access either by wireless, through a base station **26** as shown, or be hard wired to the network **14**. Wireless connections of telephones and computers to the network **14** are by means of patchpoints (PP) **29**. Redundant illumination, as shown with respect to computer **28** is preferred where possible. Also, telephones may have both wireless and hard wire access as shown with respect to location **30**.

These arrangements are shown simply as examples, and it would be clear to one skilled in the art that many alternative arrangements are also possible.

The preferred embodiment of the invention will be described firstly with regard to filters and ports, which will provide a basis for describing the major components of the invention. Those major components will then be described, followed by several implementation examples. Finally, a listing of filters and considerations for their implementation will then be presented.

In the preferred embodiment, the invention will be implemented using a distributed operating system. The software layer of the invention is independent of the hardware layer, allowing filter nodes and other software components to be located anywhere accessible in the system, dynamically mapped onto appropriate hardware, and executed. In some cases, software interfaces may be necessary to allow generalized mapping.

Filters and Ports

In the preferred embodiment, the architecture of each filter node **32** may be described as shown in **Figure 3**. In this embodiment, the function body **34** contains the executable software code, while the list of function properties **36** describes the role of the filter node **32** and does not contain executable code. This list of function properties **36** is used by users to identify the filter nodes **32** they desire, and to derive an understanding of how to apply them.

Each input and output, to or from, a filter node **32** is described as a port. Each port in the port list **38** for a given node **32**, will have an associated port name **40** and list of port properties **42**.

It is preferred that all filter nodes **32** use a standard set of input and output ports **40**, so that there is theoretical interconnectability between filter nodes **32**. This simplifies the interconnection between ports **40** at run time, speeding up execution. Ports which are not pertinent to the function of a filter node **32** may still have to be processed, possibly to coordinate the timing of the outputs. For example, a given filter node **32** may only be operating on data coming in on half of the ports **40**, but the data simply passing through the filter node **32** may have be delayed to maintain the timing integrity of the system.

This arrangement also allows new filter nodes **32** to be designed, which are not backward compatible, by incorporating new ports **40**.

Not all filter nodes **32** may be accessed by all parties. Some parties will have administrative authority, while others will have special access packages, and others, basic packages. For example, service providers may have a set of filters which they allow their subscribers to access. As well, service providers may have a set of filters to access long distance providers with which they have special business relationships.

As well, the distinction should be made between subgraphs which represent the interests of the end user, which is typically concerned primarily with the logical structure and only indirectly, through its effects on cost and performance, with the mapping onto hardware, and software that represents the interests of the service provider, which is concerned with sharing hardware resources efficiently among a large number of users. This is a server/client relationship.

Examples of filters nodes **32** are listed hereinafter, but may include:

- interfaces to different transport media such as PSTN, ATM and ADSL;
- interfaces to specific providers of long distance and other services;
- interfaces to specific hardware such as PBXs, traditional telephones and TAPI based equipment;

  various encryption techniques;
- various error correction techniques;

  features such as splitters, mergers and translators; and

  user services such as monitoring quality of service (QoS), time and use.

Ports may have a hierarchical structure. Handshaking or back pressure signals, for example, may be associated with a data stream.

Ports usually have a direction to them (input or output), although they may have components that go in the opposite direction, as for example (again) when a handshake is involved.

Examples of port types can be rationalized from the list of filter nodes **32** hereinafter, but include generally:

- sampled representations of audio signals such as linear, A-law, ADPCM, samples of pre-emphasized signals. Ports of these types are also parametrized by sample rate, number of bits, and the characteristics of pre-emphasis filters;
- coded representations of signals such as codebook-excited LPC (linear prediction coder). These can usually be parameterized, for example, by filter length and frame and sample rate;
- alerts, which signal the occurrence of an event such as a hang-up or detection of DTMF, and reset ports
- billing ports, through which representations of money flow;
- parameter ports allow call-setup software to adjust such things as sample rates, or to read them;
- state input/output ports synchronize complementary pairs of coders and decoders; and
- IP streams, and compressed versions such as RTP (real time protocol) streams.

Strong Typing of Ports

Ports **40** should preferably be "strongly typed" to avoid the setup of meaningless connections between filter nodes **32**. For example, a voice coder that expects integer samples of a voice signal will do nothing useful if driven by the output of an FEC coder. This may also require a library of ports **40** to maintain the generalized intercompatibility between filter nodes **32**. In a Java implementation, this library of ports **40** could be implemented as different interfaces.

Because the filter nodes **32** in the logical graph can represent complex computing functions, such as voice coders, only certain interconnections are valid. These nodes have properties such as latency and CPU load, for example, of interest

when wiring them up. Because the edges carry different types of information, the nodes are best thought of as having typed "ports", such as, in the voice coder case, those for linearly digitized signals and for CELP-coded (codebook excited linear predictive) voice encoding.

As an alternative to strong typing, rules of composition which specify what kinds of networks make sense, could be enforced at the Java level, or enforced by the objects themselves.

Signal Processing Object software

The Signal Processing Object software is that software that receives the graph data structure **12**, instantiates the graph and executes it. In the preferred embodiment, it will be operable to perform the functions of:

1. receiving, instantiating and executing graph data structures **12**;
2. management of filter nodes **32** and ports, including analysing, modifying, transparently adding on or choosing filters in response to inconsistencies in the received graph data structures **12**. This is not necessary in a basic system, but is desirable for the following reasons:
   a. to add filter nodes **32** to distribute billings to other service providers; or
   b. to correct small incompatibilty errors users may have made in creating graphs. For example, analogue modulated data may be transmitted over an analogue voice channel, but not a digital voice channel as the digitization will destroy the modulated data;

   The signal processing object will also insert default filter nodes **32** as necessary.
3. continuous or periodic monitoring and evaluating the services and resources available;
4. communication and negotiation with the Client agents;
5. consideration for time latencies due to processing and library access;
6. consideration for certification and privileges. Note that privileged and trusted filter nodes **32** are required for billing, OAM&P (Operations, Analysis, Maintenance and Provisioning) , NOS (Network Operating System) execution and scheduling. As well, the signal processing object should ensure that such facilities are run on trusted hardware:

Java is also widely used for programming advanced graphical user interfaces (GUIs) such as those used on some Web pages, and it is preferred that the API of the invention be written on a web-based GUI. This GUI will allow the user to inspect, modify and possibly simulate the parameters of the desired communication simply by identifying and executing icons on his operating screen. This GUI is presented to the User as a web page which may be edited using a standard browser.

The GUI provides a graphical representation on a computer screen of a collection of signal processing and input/output objects. These graphical objects corresponded directly with computer programs resident on a collection of computers specialized for digital signal processing, operating to implement the operations described by the graphical representations. The graphical representations could be interconnected by drawing lines between "ports" on them, which are labelled with names representing their functions such as "encryption" or "error correction".

Figure 4 presents the functionality of an API in the preferred embodiment of the invention. The user agent **44** allows the user to input high-level instructions to the API **46** via a GUI. These high level instructions will be received by the API **46** and processed to create and transmit graph data structures **12**. The processing will generally include the identification of filter nodes **32** which may employ library access functions **48** to obtain filter node data from remote libraries **50**, but commonly used filter node data **32** will be stored at the API **46**. The initial API **46** will be supplied to the user with standard filter nodes for basic and commonly used communications modes. If the user has frequent need for more obscure filter nodes, those may be stored on his computer or at a nearby location.

The API **46** also has access to node handling routines **52** and port manipulation and connection routines **54** which allow the assembly of graphs, and editing of filter nodes **32** as described above.

Figure 5 presents an implementation of end-to-end encryption without regard for the details of transport over the rest of the network **14** and of the processing required to get low-rate data suitable for encryption. The caller **82** assembles a graph which describes the assembly of filter nodes **32** shown in **Figure 5**, by advising the API of the called party **84**, and the desire for the communication to have a certain level of security and reliability. The API creates the graph to identify both the caller **82** and called party **84**, as well as encryption node **86** and the complementary encryption node **88** at the receiving end of the transmission. From the user's request

for a certain level of reliability, the API selects an error correction strategy and inserts nodes **89** and **90** into the graph data structure **12**.

As noted above, this graph data structure **12** may be prepared without regard for the constraints these filter nodes **32** will put on the available transmission means over the network **14**, leaving these issues to be resolved by the Signal Processing Object. When the Signal Processing Object has determined how the graph should be routed, it modifies the graph data structure **12** by inserting filter nodes **32** to route the graph data structure **12** over the desired service providers on the network **14**.

This example may be described as a two level hierarchy naturally describing the mapping from physical (less detail) to logical (more detail, because there are several computing task per computer and several logical links per T1) networks.

Not all the information flows in the signal path from one subscriber to another. There are also flows to and from the service provider's billing and management software and to the subscribers' call processing software. For example, if the number of uncorrected errors becomes excessive, it may be appropriate for the encoder to raise an exception in the call processing code so that a more robust one can be chosen. This same example also shows that it can be necessary to modify the graph while it is running.

Hiding the internals of the network **14** as in **Figure 5** is preferred most of the time, but not always. For example, a user may want to be sure that his data is never carried on a certain type of link, one belonging to a competitor, for example, or one for which availability is only statistically estimated or that it travels by two totally independent paths through the network **14** for reliability.

As noted above, it is desirable that some detail be hidden from the end user but not from the server. This would allow private business arrangements to be made between various service providers, encouraging competition and allowing lower rates and more services to be provided to users.

An exemplary arrangement of filter nodes for a voice communication over a digital network is presented in **Figure 6**. These filter nodes are described in greater detail hereinafter, but this description in given to provide an overview.

Audible communication with the user is performed with the microphone and speaker combination **96**, which is usually provided in the form of a combination handset in the case of traditional telephony, or a microphone and speaker set in the

case of a personal computer. The audio signals are processed by an acoustic echo canceller **98** which attempts to filter and reject audible reverberations.

The signal input by the user is then coded by a voice coder **100** which digitizes the analogue voice signal coming from the acoustic echo canceller **98**. The voice coder **100** also performs a complementary operation of converts the digital signal received from the encryption node **102**, into an analogue signal which is passed on to the acoustic echo canceller **98**.

The voice coder **100** also has a bi-direction communication with the encryption node **102** which encrypts the signal originated by the user, and de-crypts the incoming signal from the forward error correction encoder **104**. Forward error correction (FEC) is well known in the art, and is commonly applied in wireless communication. Briefly, it consists of adding codes to a transmitted signal to allow the recipient to detect and correct erroneous data, but at the expense of bandwidth.

The final interface is the modem (modulator/demodulator) driver **106**, which modulates the signal onto a carrier frequency for transmission by radio channel, or similar device.

Signal Path Filters

a. linear and adaptive filters

Classic linear filtering is used to remove DC and 60/120/180Hz tones from power-line interference and to smooth signals for down sampling. In a digital system the down sampling and filter are usually combined in a more efficient decimation or rate-conversion block. Other applications, standard in audio but rarer in telephony, include tone controls and generation of reverberation. Computation loads for simple filters are very small, on the order of 1-10 multiply-adds per sample (80 kIPS), and are completely predictable. If the processor on which a filter is running crashes and a new filter is restarted, there will be an audible "click" unless state is preserved, and the internal state may vary quite quickly.

The main requirement that filtering places on the system are:

- that multiply-adds at the 16-24 bit level be fast; and
- that overheads for simple algorithms be small.

Adaptive filters tune their coefficients to the particular call in progress. The best-known case in telephony is the echo canceller, such as the echo canceller **98** of **Figure 6**, of which variants are designed to cancel acoustic echoes resulting from an

2Mb/s, hence the 3G requirement for that rate. However, MPEG2 is bursty, needing more capacity when the image changes suddenly.

At the low-quality end, video conferencing is usually done at 128kb/s. At this rate the coding process adds hundreds of msec of delay and the picture is poor.

If full-motion video is demanded, then 5MHz slots will not have sufficient capacity, though 20MHz slots and generous use of antenna diversity could support 10-40 users at that rate.

f. Voice-mail and its video and text equivalents are usually thought of as pure data, but should be seen as objects with methods for reading and writing, or as filters that persist after calls complete. Generalizing allows different types of coders, including encryption and fax data, to be used in a flexible manner with voice-mail.

g. Reading voice-mail can be thought of as accepting a call, though a time-shifted one. Voice-mails are all pending requests to the called party's proxy, and display the graph of the call that set them up even though it has long since been torn down, so that the accepting party can see data type, coding and encryption needs, source of call, check who is paying, and other parameters.

A single-use proxy stays alive and attached to the mail, into which a user may also call to retrieve mail. This permits group messaging or retrieval by password, situations in which the voice-mail does not know who to contact. In this sense reading voice-mail can be thought of as originating a call.

Channel Coding

a. Forward Error Correction (FEC), such as that performed by the forward error correction encoder **104** in **Figure 6**, uses mathematical algorithms such as XOR convolutions between the data and a given sequence to produce redundant bits that can be used to detect and correct errors in transmission. For security of wireless implementations, this will be important. Automatic Repeat reQuest (ARQ) schemes like transmission control protocol (TCP) are simple and efficient and can be arbitrarily reliable, but add variable latency, making them impractical for voice. In telephony, frames which can not be corrected using FEC are discarded, and a reasonable rate of data loss is considered acceptable. There are also trade-offs with respect to redundancy, power and error rate.

though triple data encryption standard (DES), standard in the banking industry, is not too unmanageable, requiring a couple of dozen bit-shuffles and O(100) 4-bit table lookups.

Internet traffic will best be encoded in the user's PC rather than at patch points **29**, to minimize the amount of cleartext running around. Still, if the invention is being used to implement a virtual private network the users may be transmitting cleartext around their Ethernets and using the invention to bridge remote Ethernets, in which case encryption should be performed at the patch points to provide a secure solution.

It is also preferred to use good encryption, including signature techniques, on wireless control links.

Modems

As shown in the system diagram of **Figure 2**, users are intended to have a data connection to the network **14**, though they may also have a PSTN RJ-11 connection. For Internet use, the data (Ethernet) connection would be faster, though for fax use, the implementation might be simplified by use of the PSTN RJ-11 connection. A PSTN landline could be simulated over the data network **14** to accommodate a fax connection, perhaps using adaptive delta pulse code modulation (ADPCM), or even straight pulse code modulation (PCM), but a voice coder would have to avoided as it would destroy the data.

The preferred solution is to detect that the data source is a fax, and implement a fax modem in software at the patch point **29** and at the gateway closest to the receiver so that only the raw fax data need be transmitted. This can be used to economize on latency as well as on bandwidth, but requires that the fax at the receiving end can be spoofed by the local modem. An exemplary modem application is presented in **Figure 6** by means of modem filter **106**.

IP fax is seen in the industry as a desirable feature, because the volume of fax traffic currently rivals that of voice traffic.

Voice Mail

Current telephony practise for voice-mail is to convert calls to high-rate PCM, transmit the converted voice messages to a point near to intended recipient at high

bandwidth and with the usual telephony low latency, then voice-code them to save space and store them on disk.

Network load can be reduced by leaving calls coded as for the wireless link, and by accepting long latencies for coded packets, for example, using best-effort service. This also allows the encrypted voice data to be left encrypted in transit and storage. Disk space may be used wherever it is available, though it is best to move it in advance close to the most likely place from which it will be read because latency is less tolerated when listening to voice-mail.

Integration of a user's various mailboxes including e-mail, voice-mail and fax, is a current industry trend and is a desirable utility for the invention.

Links Between the Signal and Control Paths

a. Dual tone multi-frequency (DTMF) signalling is the familiar "touch-tone" technique of simultaneously transmitting a pair of tones each at one of four frequencies to signal switches for dialling and end-user equipment for voice-mail. The DTMF encoders can be implemented with a simple filter or a table look-up arrangement, and the DTMF receivers can be implemented with a group of filters and slicers at roughly 30-100 operations/sample.

b. Pulse dialling detection involves counting strings of open-circuits on the telephone line at about 10Hz. The "flash" or "link" buttons often used to signal the desire to set up a 3-way call, basically dial "1".

Both the pulse dialling and DTMF filters described above provide inputs to call processing software from the signal path. That path does not have tight latency requirements, unless the user wishes to suppress the DTMF signal in the path to a called party.

c. Voice recognition can be used to replace dialling and to offer more sophisticated call control from a conventional telephone, or to do authentication. Computational loads can be quite large, larger than voice coding, for example, which is typically a component of voice recognition, and the area is still subject to active research. Computational loads can also vary widely as a function of the input data.

Ideally, voice-operated services can be speaker-independent, but systems that can handle large vocabularies generally have to be trained for the speaker. As

well, voice authentication systems require training of the receiving filter. The need for speaker-dependent state suggests that these algorithms will need access to a library, which they may also want to update if they are capable of continual retraining.

The heavy loads, the use of disk resources, and the fact that voice coding is typically the first step combine to suggest that a typical voice recognition application would do voice coding on the patch point **29** but might do the rest of the numerical processing on a computer server at the base station **26** side. The optimal voice coder for voice recognition is not necessarily the optimal or standard one for wireless, so choosing to use voice recognition somewhere in a call graph may constrain the type of voice coder used elsewhere. The call setup agent could just ship both kinds of data, but that would be an expensive use of the wireless resource.

d. Call progress tones like dial-tone and ring and busy signals give call-processing software a way to drive the signal path. Modern systems also allow the use of speech clips, though these are of questionable utility in multilingual environments. A language preference may be designated as part of a user's state.

Users who have a good display device available may prefer to use it rather than to hear ringing tones. This is an interesting example of the need to be able to abstract part of call processing. It is preferred to be able to "plug in" arbitrary ways of notifying the end user that a line is busy without changing the rest of a piece of call processing software. This would also allow easy customizing of audible signals, such as the use of a Beethoven sound bite, rather than a traditional single tone announcement.

IP packets to and from the Ethernet port

It is preferred to be able to filter IP packets that come in from the Ethernet port, too. There is not necessarily any warning that IP packets are coming, since IP is connectionless, but that just means that IP filtering is set up by default.

a. IP classifiers assign different types of traffic different priorities, taking a single input, unclassified IP, and producing multiple output streams. Packets should be classified before they are transmitted over the expensive wireless link so as to implement the user's own policies on what to pay premium rates for. A default classifier may, for example, assign a lower priority to Web traffic than to IP telephony,

or give one particular Ethernet source higher priority than others. Classifiers belong at the network input and perhaps also at both ends of the wireless link.

IP classifiers can also manage traffic in the other direction, regulating flows onto the Ethernet from the patch point **29**.

b. Traffic shaping, traffic policing and radio resource management go together for IP packets. Traffic shaping typically uses a leaky bucket strategy to force traffic statistics to match the profile promised in a quality of service (QoS) negotiation.

A "leaky bucket" is a technique used in ATM and RSVP to specify average bandwidth. Traffic is modelled in terms of the average output rate and the size of the input buffer needed to smooth bursts out to that rate. A long burst will overflow the bucket, and packets that overflow the bucket are typically marked as candidates for deletion if the network overloads.

For the wireless link these parameters might be interpreted literally, allocating enough radio slots/channels to handle the rate, and putting a buffer at the sending side. For an optical link it may be interpreted only as a specification that defines which packets may be marked for sacrifice. A variant mechanism is a "token bucket" that allows bursts at full speed until the flow has used up a bucket full of tokens, then restricts flow rate to the required average as tokens dribble in. These mechanisms directly express queueing behaviour, which is fundamental to networking, so they are the preferred ones to use.

"QoS Negotiation" refers to a technique described in greater detail in the co-pending patent application under the Patent Cooperation Treaty, Serial No. _____, titled "Method and System for Negotiating Telecommunication Resources". Briefly, a calling party creates a graph including a desired QoS and proposed pricing, and transmits it to the service provider for consideration. The service provider may accept the proposal, issue a counter proposal, or abandon the negotiation.

For new services, one cannot assume that there is a single pipe at a given bandwidth and quality of service (QoS) involved in a call. For example, a 3-way video-conference call might have one of the branches operating as voice-only at a much lower rate than the video branches. The API for negotiation has to capture the whole structure of the call. For this reason, and to avoid adding new constructs, the

ii) technical issues, such as whether caching makes cookie-based pages malfunction; and

iii) business issues such as whether it makes advertising banners and referrals miscount.

Implementation of caching has clear performance gains but application should give consideration to these issues.

g. Packet formatting and assembly of data for transmission over ATM and other transport protocols. There are presently a number of such transport media with specifications generally known in the art. Design of packet formatting filters conforming to the protocol of the particular transport medium would be within the skill of one in the art, in view of the teachings of the invention.

h. Meta tags

i. Interfacing with "socket" mechanisms such as H.323. As noted above, H.323 is widely used to describe connections between applications programs running on operating systems such as UNIX and Windows. It can be used to set up connections between applications programs running on different computers, such that packets of data are passed between them across such networks as an Ethernet or the Internet. In Java, for example, the expression new Socket("www.wireless-sys.com", 8888) returns an object that represents a connection to "port 8888" on a computer on the Internet whose name is "www.wireless-sys.com". This object can be used with other Java methods to send data to, and receive data from, this computer. The "port number" is used by convention to define the type of data expected.

When using a socket to communicate with a process on another computer, the programmer defines one side of a communication but must rely on the administrators of the other computer to have set up the other side. The port number is used by convention to describe the functionality of the program expected.

Sockets typically use the Internet Protocol (IP) and can further be set up to use either the "unreliable datagram protocol", UDP, which sends packets without checking to see if they have been received, or the Transport Control Protocol (check), TCP, which will retry until it receives a confirmation of receipt. Telephony applications typically use UDP, because data that does not arrive on time is of no

3. A link may reflect the characteristics of the service desired by the user. These characteristics may include required network bandwidth and quality of service parameters such as delay, reliability, error rate, or packet loss probability.

4. A link may reflect the characteristics of the service to which that the proxies for the user and the service provider have agreed. These characteristics may include allocated network bandwidth and quality of service parameters such as delay, error rate, or packet loss probability.

5. A link may reflect the physical delay in the signaling medium (e.g., wire, fibre, wireless). It may additionally reflect the queueing delay experienced by a packet that has been routed through the link.

6. A link may include costing parameters or a costing formula by which the service provider can obtain a quote for the total cost associated with the filter graph.

7. A link may be annotated with parameters that specify the billing mechanisms to be used.

8. A link may be annotated with information necessary to interact with an OAM&P system using SNMP.

While particular embodiments of the present invention have been shown and described, it is clear that changes and modifications may be made to such embodiments without departing from the true scope and spirit of the invention. For example, one could implement the invention without a distributed operating system, by hardcoding the locations of the filter nodes into any graph structures and still realize many of the benefits of the invention.

The method steps of the invention may be embodiment in sets of executable machine code stored in a variety of formats such as object code or source code. Such code is described generically herein as programming code, or a computer program for simplification. Clearly, the executable machine code may be integrated with the code of other programs, implemented as subroutines, by external program calls or by other techniques as known in the art.

The embodiments of the invention may be executed by a computer processor or similar device programmed in the manner of method steps, or may be executed by an electronic system which is provided with means for executing these steps.

Similarly, an electronic memory medium such a computer diskette, CD-Rom, Random Access Memory (RAM), Read Only Memory (ROM) or similar computer software storage media as known in the art, may be programmed to execute such method steps. Further, electronic signals representing these method steps may also be transmitted via a communication network such as the Internet.

It would also be clear to one skilled in the art that this invention need not be limited to the existing scope of computers and computer systems. Any telecommunication system could employ broad aspects of the invention including radio systems, television broadcasting, satellite communications, bank automated tellers, point of sale computers, local area networks and wide area networks. A point of sale computer, for example, may run almost all the time in a certain mode, but be accessed remotely to download sales data or update pricing. Again, such implementations would be clear to one skilled in the art, and do not take away from the invention.

Finally, numerous modifications, variations, and adaptations may be made to the particular embodiments of the invention described above without departing from the scope of the invention.

WHAT IS CLAIMED IS:

1. A method of implementing a communication over a telecommunications network comprising the steps of:
composing said communication in terms of a graph of software building blocks; and
dynamically instantiating said graph of software building blocks at run time.

2. A method as claimed in claim 1, comprising the subsequent step of executing said instantiated software building blocks.

3. A method as claimed in claim 2, wherein said communication is between a first party and a second party interconnected via said telecommunications network, and said steps of composing and dynamically instantiating comprise the steps of:
composing said communication in terms of a graph of filter nodes;
dynamically instantiating at run time, said graph of filter nodes; and
dynamically configuring at run time, each of said filter nodes.

4. A method as claimed in claim 3, wherein said step of executing said instantiated software building blocks includes executing said instantiated software building blocks on said telecommunications network.

5. A method as claimed in claim 4, wherein said step of composing comprises:
composing said communication as a graph of filter nodes, each said filter having an associated set of properties and input/output ports.

6. A method as claimed in claim 6, wherein prior to said step of dynamically instantiating, performing the step of:
identifying locations of said filter nodes in a distributed operating system.

7. A method as claimed in claim 6, further comprising the step of:
strong typing by analysing filter properties and verifying that connections are meaningful.

8. A method as claimed in claim 7, wherein said step of composing comprises:
composing said communication as a graph of filter nodes accessible to said first or second parties or said telecommunications network, each said filter having an associated set of properties and input/output ports, said input/output ports being standardized for all filters.

9. A method as claimed in claim 8, further comprising the step of:
responding to a request for execution of an identified filter node only to authorized parties.

10. A method as claimed in claim 9, further comprising the step of:
responding to request for modification of an identified filter node only to authorized parties.

11. A method of implementing an application programming interface (API) for graph-based implementation of telecommunications comprising the steps of:
receiving input instructions; and
responding to said input instructions by generating a graph describing the desired functionality of said communication.

12. A method as claimed in claim 11, further comprising the step of:
responding to a request to add a filter node to a communication by:
determining the configuration of said filter node with respect to other filter nodes defined in said graph; and
storing said configuration along with the identification of said filter node, in said graph.

13. A method as claimed in claim 12, further comprising the step of:
responding to a request to test a generated graph by:
analysing the filter nodes of said graph, and their associated configuration, properties and ports;
simulating execution of said graph; and
providing output data indicating the results of said graph test.

14. A method as claimed in claim 13, wherein:
said step of receiving input instructions comprises responding to identification of icons in a graphic user interface (GUI); and
said step of providing output data comprises providing output data to said GUI in a graphic format.

15. A method as claimed in claim 14, wherein said step of receiving input instructions comprises:
responding to identification of graphic icons in a graphic user interface (GUI) presented as a web page which may be edited using a standard web browser.

16. A method of implementing a communication over a telecommunications network, said communication being defined in terms of a graph of software building blocks, said method comprising the steps of:
dynamically instantiating said graph of software building blocks at run time.

17. A method as claimed in claim 16, comprising the subsequent step of executing said instantiated software building blocks.

18. A method as claimed in claim 17, wherein said communication is between a first party and a second party interconnected via said telecommunications network, and said communication is defined in terms of a graph of filter nodes, and wherein said step of dynamically instantiating comprises the steps of:
dynamically instantiating at run time, said graph of filter nodes; and
dynamically configuring at run time, each of said filter nodes.

19. A method as claimed in claim 18, wherein said step of executing said instantiated software building blocks includes executing said instantiated software building blocks on said telecommunications network.

20. A method as claimed in claim 19, further comprising the step of:
identifying the location of each of said filter nodes.

21. A method as claimed in claim 20, further comprising the step of:

responding to an inconsistency between two filter nodes in said graph by modifying one of said filter nodes or the ports of one of said filter nodes to resolve said inconsistency.

22. A method as claimed in claim 21, further comprising the step of:

responding to an inconsistency between two filter nodes in said graph by:

identifying an appropriate correcting filter node; and

transparently interconnecting ports of said correcting filter node to those of the adjacent filter nodes.

23. A computer data signal embodied in a carrier wave, said computer data signal comprising a set of machine executable code being executable by a computer to perform the steps of any one of claims 1, 11 or 16.

24. A computer readable storage medium storing a set of machine executable code, said set of machine executable code being executable by a computer server to perform the steps of any one of claims 1, 11 or 16.

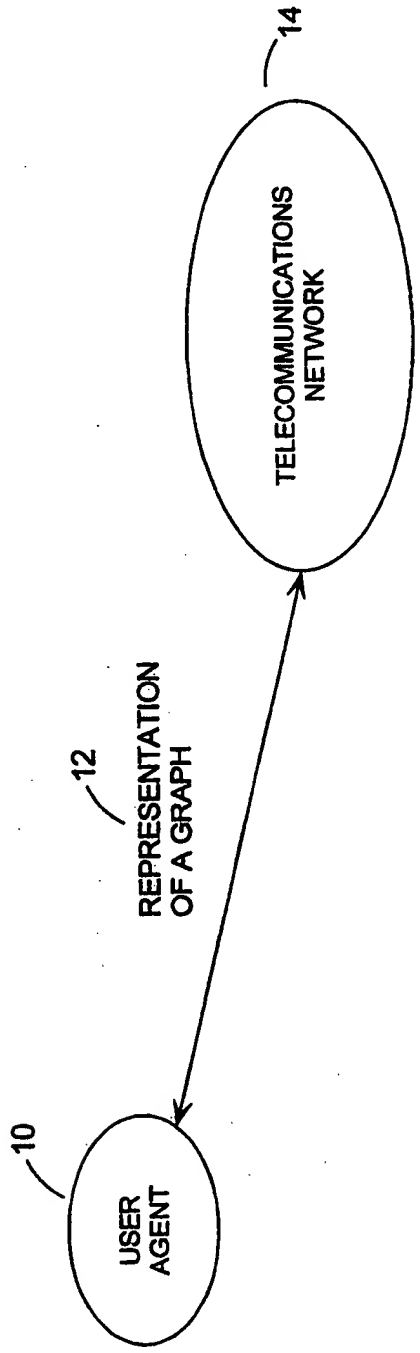
14
TELECOMMUNICATIONS NETWORK
12
REPRESENTATION OF A GRAPH
10
USER AGENT

FIGURE 1

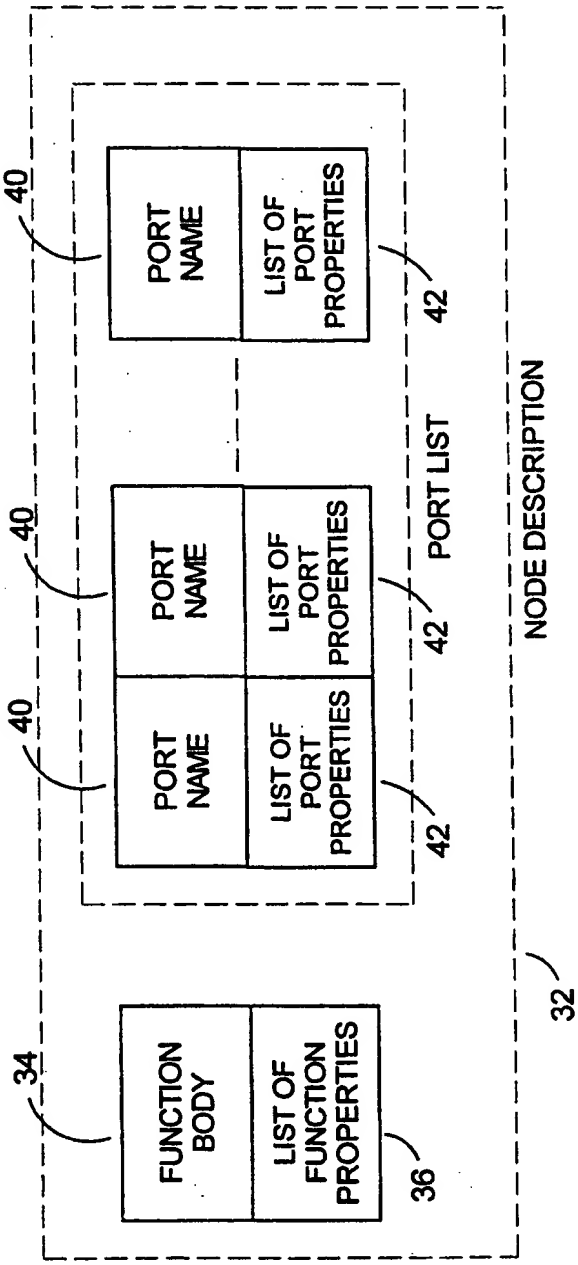
NODE DESCRIPTION
32
PORT LIST
PORT NAME
LIST OF PORT PROPERTIES
40
42
FUNCTION BODY
LIST OF FUNCTION PROPERTIES
34
36

FIGURE 3

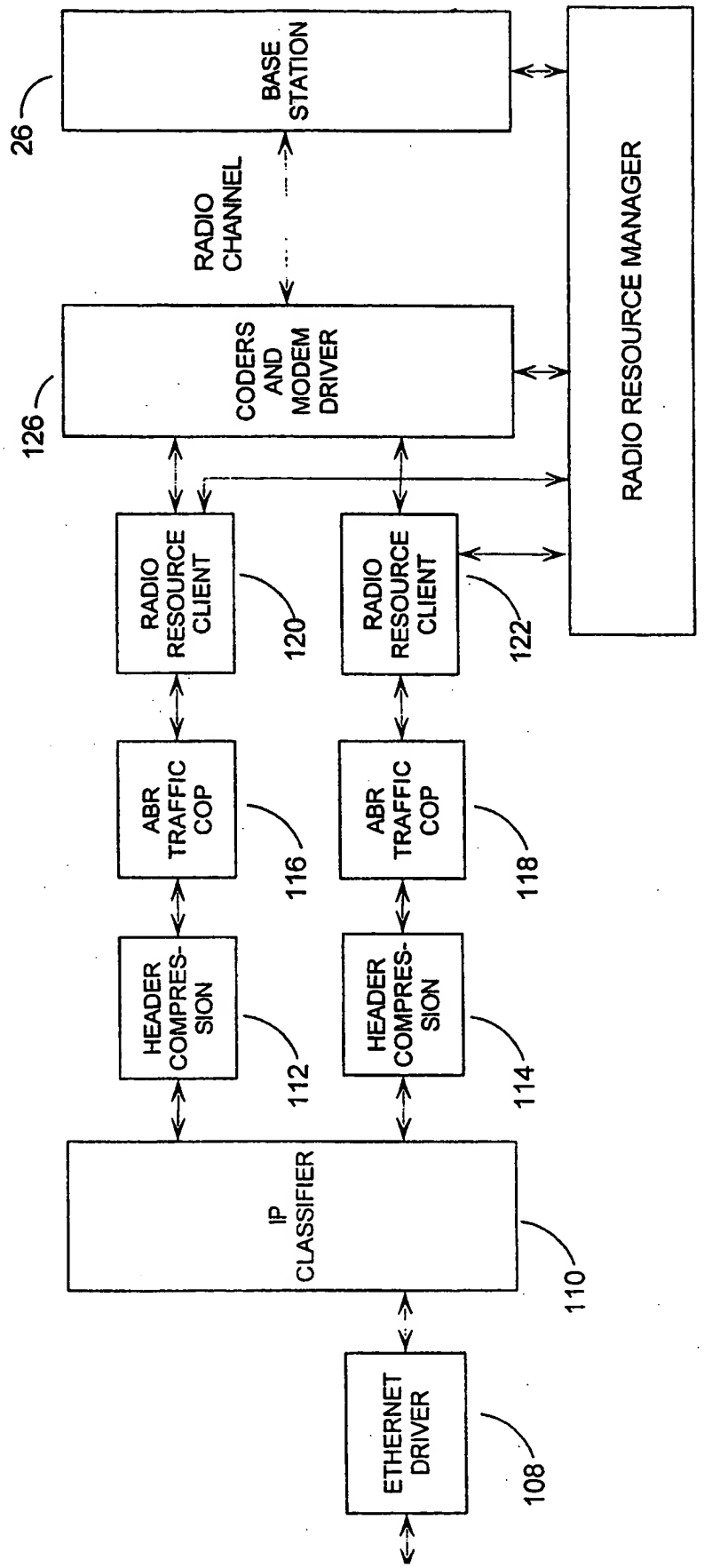
BASE STATION
26
RADIO CHANNEL
CODERS AND MODEM DRIVER
126
RADIO RESOURCE MANAGER
124
RADIO RESOURCE CLIENT
120
RADIO RESOURCE CLIENT
122
ABR TRAFFIC COP
116
ABR TRAFFIC COP
118
HEADER COMPRES-SION
112
HEADER COMPRES-SION
114
IP CLASSIFIER
110
ETHERNET DRIVER
108

FIGURE 7